Shakin’ Hands and Living in SYN: A TCP Tale

CSE 461 Section
Joke Later!

- Let’s learn things first!
TCP Is Reliable

• What do we mean by “reliable?”
  • We know when the other party receives or doesn’t receive certain data
  • Data arrives intact
  • Data arrives in the correct order (to the application layer, at least)
What’s the main mechanism for ensuring this reliability?

- Sequence numbers!
- They allow packets to be identified, acknowledged, and implicitly re-requested
- For TCP to work, clients must know each other’s sequence number schemes
Starting Communication: The Three-Way Handshake

- Need to synchronize with each other’s sequence numbers
- How can we do this?
- Active open vs. passive open
  - connect() vs. listen()
- SYN packet
  - Send own sequence number A
- SYN/ACK packet
  - Acknowledge with A+1, send own sequence number B
- ACK packet
  - Acknowledge with B+1
- Demonstration
Three-Way Handshake Diagram

- **Initiator**
  - `connect()`
  - SYN
  - SYN-ACK
  - ACK
  - Success code returned by `connect()`
  - (Data packets exchanged)

- **Listener**
  - `listen()`
  - TCB initialized to SYN-RECEIVED state
  - TCB transitions to ESTABLISHED state
Ending Communications

- We need a protocol for stopping communications
- What could we do?
- Let’s send packets to close the connection!
- FIN/ACK sequence
TCP Half-Open

- TCP Half-Open
  - One client is in the open state; the other is not
- How could this happen?
  - One endpoint has crashed
  - One endpoint has removed the socket
  - One endpoint has received a SYN and sent a SYN/ACK, but the other side has not ACKed the SYN/ACK yet
  - One endpoint has sent a FIN and received an ACK, but the other side has not sent a FIN yet
  - RST packet often sent in these cases
Hosts allocate a TCB (Transmission Control Block) on receipt of a SYN packet and put it in the "SYN queue".

They then send a SYN/ACK and wait for an ACK response.

However, what if one host sends lots of SYN packets and no ACKs? This causes the receiving host to allocate lots of TCBs, filling up resources. This attack is called "SYN flooding".
SYN Flooding Countermeasures

• What ideas can we think of to make it so that SYN flooding doesn’t work?
• Constraint: we don’t want to break TCP!
• Identify SYN flooders and filter their packets
• Reduce our timeout until we garbage-collect TCBs
• Recycle half-open TCP connections
• Use SYN cookies
• Sequence number encodes all of the data that would otherwise be stored
• This allows us to garbage-collect our SYN queue and still respond to subsequent ACKs
TCP Connection Hijacking

- TCP is not (by default) encrypted
- This means anyone sniffing our packets can see the sequence numbers being used
- How is this a problem?
  - For many protocols, the sequence and acknowledgement numbers are the other “security”
  - Using these numbers can make a host think that you’re sending the next packet in a communication session
  - This can cause the communication to be re-addressed to a new IP address/port
TCP Veto

- In TCP, how does a server know to discard a duplicate packet? What does it check for?
  - Correct checksum
  - Same sequence number
- How are sequence numbers generated?
  - Randomly at first, then incremented
  - Often, this increment is unpredictable, and depends on received data length
- How could we secret inject a packet into communication?
  - Predict the length and sequence number of some data in the future
  - Pre-empt that data with a similar packet
Joke Time

• Two jokes
Questions?